Particle swarm optimization

From Scholarpedia

Marco Dorigo et al. (2008), Scholarpedia, 3(11):1486. revision #52052 [link to/cite this article]

Curator: Dr. Marco Dorigo, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
Curator: Mr. Marco A. Montes de Oca, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
Curator: Prof. Andries Engelbrecht, Department of Computer Science, University of Pretoria

Particle swarm optimization (PSO) is a population-based stochastic approach for solving continuous and discrete optimization problems.

In particle swarm optimization, simple software agents, called *particles*, move in the search space of an optimization problem. The position of a particle represents a candidate solution to the optimization problem at hand. Each particle searches for better positions in the search space by changing its velocity according to rules originally inspired by behavioral models of bird flocking.

Particle swarm optimization belongs to the class of swarm intelligence techniques that are used to solve optimization problems.

Contents

History

Particle swarm optimization was introduced by Kennedy and Eberhart (1995). It has roots in the simulation of social behaviors using tools and ideas taken from computer graphics and social psychology research.

Within the field of computer graphics, the first antecedents of particle swarm optimization can be traced back to the work of Reeves (1983), who proposed particle systems to model objects that are dynamic and cannot be easily represented by polygons or surfaces. Examples of such objects are fire, smoke, water and clouds. In these systems, particles are independent of each other and their movements are governed by a set of rules. Some years later, Reynolds (1987) used a particle system to simulate the collective behavior of a flock of birds. In a similar kind of simulation, Heppner and Grenander (1990) included a *roost* that was attractive to the simulated birds. Both models inspired the set of rules that were later used in the original particle swarm optimization algorithm.

Social psychology research, in particular the dynamic theory of social impact (Nowak, Szamrej & Latané, 1990), was another source of inspiration in the development of the first particle swarm optimization algorithm (Kennedy, 2006). The rules that govern the movement of the particles in a problem's search space can also be seen as a model of human social behavior in which individuals adjust their beliefs and attitudes to conform with those of their peers (Kennedy & Eberhart 1995).

Standard PSO algorithm

Preliminaries

The problem of minimizing[1] the function $f : \Theta \to \mathbb{R}$ with $\Theta \subseteq \mathbb{R}^n$ can be stated as finding the set

$$\Theta^* = \underset{\vec{\theta}\in\Theta}{\arg\min}\, f(\vec{\theta}) = \{\vec{\theta}^* \in \Theta \colon f(\vec{\theta}^*) \le f(\vec{\theta}), \quad \forall \vec{\theta} \in \Theta\},$$

where $\vec{\theta}$ is an n-dimensional vector that belongs to the set of feasible solutions Θ (also called search space).

In PSO, the so-called *swarm* is composed of a set of particles $\mathcal{P} = \{p_1, p_2, \ldots, p_k\}$. The position of a particle corresponds to a candidate solution of the considered optimization problem, which is represented by an objective function f. At any time step t, p_i has a position $\vec{x}_i^{\,t}$ and a velocity $\vec{v}_i^{\,t}$ associated to it. The best position that particle p_i (with respect to f) has ever visited until time step t is represented by vector $\vec{b}_i^{\,t}$ (also known as a particle's *personal best*). Moreover, a particle p_i receives information from its *neighborhood* $\mathcal{N}_i \subseteq \mathcal{P}$. In the standard particle swarm optimization algorithm, the neighborhood relations between particles are commonly represented as a graph $G = \{V, E\}$, where each vertex in V corresponds to a particle in the swarm and each edge in E establishes a neighbor relation between a pair of particles. The resulting graph is commonly referred to as the swarm's *population topology* (Figure 1).

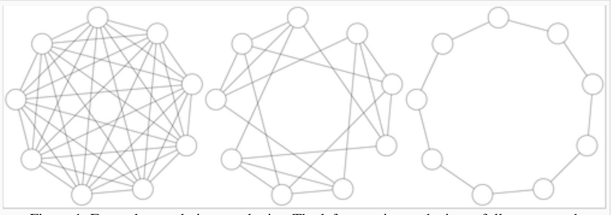
Figure 1: Example population topologies. The leftmost picture depicts a fully connected topology, that is, $\mathcal{N}_i = \mathcal{P}\ \forall p_i \in \mathcal{P}$ (self-links are not drawn for simplicity). The picture in the center depicts a so-called von Neumann topology, in which $|\mathcal{N}_i| = 4\ \forall p_i \in \mathcal{P}$. The rightmost picture depicts a ring topology in which each particle is neighbor to two other particles.

The algorithm

The PSO algorithm starts by generating random positions for the particles, within an initialization region $\Theta' \subseteq \Theta$. Velocities are usually initialized within Θ' but they can also be initialized to zero or to small random values to prevent particles from leaving the search space during the first iterations. During the main loop of the algorithm, the velocities and positions of the the particles are iteratively updated until a stopping criterion is met.

The update rules are:

$$\vec{v}_i^{\,t+1} = w\vec{v}_i^{\,t} + \varphi_2 \vec{U}_1^{\,t}(\vec{b}_i^{\,t} - \vec{x}_i^{\,t}) + \varphi_2 \vec{U}_2^{\,t}(\vec{l}_i^{\,t} - \vec{x}_i^{\,t}),$$

$$\vec{x}_i^{\,t+1} = \vec{x}_i^{\,t} + \vec{v}_i^{\,t+1},$$

where w is a parameter called *inertia weight*, φ_1 and φ_2 are two parameters called *acceleration coefficients*, $\vec{U}_1^{\,t}$ and $\vec{U}_2^{\,t}$ are two $n \times n$ diagonal matrices in which the entries in the main diagonal are random numbers uniformly distributed in the interval $[0, 1)$. At each iteration, these matrices are regenerated. Usually, vector $\vec{l}_i^{\,t}$, referred to as the *neighborhood best,* is the best position ever found by any particle in the neighborhood of particle p_i, that is, $f(\vec{l}_i^{\,t}) \le f(\vec{b}_j^{\,t})\ \forall p_j \in \mathcal{N}_i$. If the values of w, φ_1 and φ_2 are properly chosen, it is guaranteed that the particles' velocities do not grow to infinity (Clerc and Kennedy 2002).

The three terms in the velocity-update rule above characterize the local behaviors that particles follow. The first term, called the *inertia* or *momentum* serves as a memory of the previous flight direction, preventing the particle from drastically changing direction. The second term, called the *cognitive component* models the tendency of particles to return to previously found best positions. The third term, called the *social component* quantifies the performance of a particle relative to its neighbors. It represents a group norm or standard that should be attained.

In some cases, particles can be attracted to regions outside the feasible search space Θ. For this reason, mechanisms for preserving solution feasibility and a proper swarm operation have been devised (Engelbrecht 2005). One of the least disruptive mechanisms for preserving feasibility is one in which particles going outside Θ are not allowed to improve their personal best position so that they are attracted back to the feasible space in subsequent iterations.

A pseudocode version of the standard PSO algorithm is shown below:

```
:Inputs Objective function $f : \Theta \to \mathbb{R}$ , the initialization domain $\Theta' \subseteq \Theta$,
the number of particles $|\mathcal{P}| = k$ , the parameters $w$ , $\varphi_1$, and $\varphi_2$, and the stopping criterion $S$
:Output Best solution found

// Initialization
Set t := 0
for i := 1 to k do
   Initialize $\mathcal{N}_i$ to a subset of $\mathcal{P}$ according to the desired topology
   Initialize $\vec{x}_i^t$ randomly within $\Theta'$
   Initialize $\vec{v}_i^t$ to zero or a small random value
   Set $\vec{b}_i^t = \vec{x}_i^t$
end for

// Main loop
while $S$ is not satisfied do

   // Velocity and position update loop
   for i := 1 to k do

      Set $\vec{l}_i^t$ := $\underset{\vec{b}_j^t \in \Theta \,|\, p_j \in \mathcal{N}_i}{\arg\min} \; f(\vec{b}_j^t)$
      Generate random matrices $\vec{U}_1^t$ and $\vec{U}_2^t$
      Set $\vec{v}_i^{t+1}$ := $w\vec{v}_i^t + \varphi_1\vec{U}_1^t(\vec{b}_i^t - \vec{x}_i^t) + \varphi_2\vec{U}_2^t(\vec{l}_i^t - \vec{x}_i^t)$
      Set $\vec{x}_i^{t+1}$ := $\vec{x}_i^t + \vec{v}_i^{t+1}$
   end for

   // Solution update loop
   for i := 1 to k do
      if $f(\vec{x}_i^t) < f(\vec{b}_i^t)$
         Set $\vec{b}_i^t$ := $\vec{x}_i^t$
      end if
   end for

   Set t := t + 1

end while
```

The algorithm above implements synchronous updates of particle positions and best positions, where the best position found is updated only after all particle positions and personal best positions have been updated. In asynchronous update mode, the best position found is updated immediately after each particle's position update. Asynchronous updates lead to a faster propagation of the best solutions through the swarm.

Main PSO variants

The original particle swarm optimization algorithm has undergone a number of changes since it was first proposed. Most of these changes affect the way the velocity of a particle is updated. In the following subsections, we briefly describe some of the most important developments. For a more detailed description of many of the existing particle swarm optimization variants, see (Kennedy and Eberhart 2001, Engelbrecht 2005, Clerc 2006 and Poli et al. 2007).

Discrete PSO

Most particle swarm optimization algorithms are designed to search in continuous domains. However, there are a number of variants that operate in discrete spaces. The first variant proposed for discrete domains was the binary particle swarm optimization algorithm (Kennedy and Eberhart 1997). In this algorithm, a particle's position is discrete but its velocity is continuous. The j th component of a particle's velocity vector is used to compute the probability with which the j th component of the particle's position vector takes a value of 1. Velocities are updated as in the standard PSO algorithm, but positions are updated using the following rule:

$$x_{ij}^{t+1} = \begin{cases} 1 & \text{if } r < sig(v_{ij}^{t+1}), \\ 0 & \text{otherwise,} \end{cases}$$

where x_{ij} is the j th component of the position vector of particle p_i , r is a uniformly distributed random number in the interval $[0,1)$ and

$$sig(x) = \frac{1}{1+e^{-x}} \,.$$

Constriction Coefficient

The *constriction coefficient* was introduced as an outcome of a theoretical analysis of swarm dynamics (Clerc and Kennedy 2002). Velocities are constricted, with the following change in the velocity update: $\vec{v}_i^{t+1} = \chi^t[\vec{v}_i^t + \varphi_2\vec{U}_1^t(\vec{b}_i^t - \vec{x}_i^t) + \varphi_2\vec{U}_2^t(\vec{l}_i^t - \vec{x}_i^t)]$, where χ^t is an $n \times n$ diagonal matrix in which the entries in the main diagonal are calculated as

$\chi_{jj}^t = \dfrac{2\kappa}{|2 - \varphi_{jj}^t - \sqrt{\varphi_{jj}^t(\varphi_{jj}^t - 2)}|}$ with $\varphi_{jj}^t = \varphi_1 U_{1,jj}^t + \varphi_2 U_{2,jj}^t$. Convergence is guaranteed under the conditions that $\varphi_{jj}^t \geq 4\,\forall j$ and $\kappa \in [0,1]$.

Bare-bones PSO

The *bare-bones particle swarm* (Kennedy 2003) is a version of the particle swarm optimization algorithm in which the velocity- and position-update rules are substituted by a procedure that samples a parametric probability density function.

In the bare-bones particle swarm optimization algorithm, a particle's position update rule in the j th component is $x_{ij}^{t+1} = N\left(\mu_{ij}^t, \sigma_{ij}^t\right)$, where N is a normal distribution with

$$\begin{aligned} \mu_{ij}^t &= \tfrac{b_{ij}^t + l_{ij}^t}{2}, \\ \sigma_{ij}^t &= |b_{ij}^t - l_{ij}^t| \,. \end{aligned}$$

Fully informed PSO

In the standard particle swarm optimization algorithm, a particle is attracted toward its best neighbor. A variant in which a particle uses the information provided by all its neighbors in order to update its velocity is called the *fully informed particle swarm* (FIPS) (Mendes et al. 2004).

In FIPS, the velocity-update rule is

$$\vec{v}_i^{t+1} = w\vec{v}_i^{t} + \frac{\varphi}{|\mathcal{N}_i|}\sum_{p_j \in \mathcal{N}_i} \mathcal{W}(\vec{b}_j^{t})\vec{U}_j^{t}(\vec{b}_j^{t} - \vec{x}_i^{t}) ,$$

where $\mathcal{W} : \Theta \to [0,1]$ is a function that weighs the contribution of a particle's personal best position to the movement of the target particle based on its relative quality.

Applications of PSO and Current Trends

The first practical application of PSO was in the field of neural network training and was reported together with the algorithm itself (Kennedy and Eberhart 1995). Many more areas of application have been explored ever since, including telecommunications, control, data mining, design, combinatorial optimization, power systems, signal processing, and many others. To date, there are hundreds of publications reporting applications of particle swarm optimization algorithms. For a review, see (Poli 2008). Although PSO has been used mainly to solve unconstrained, single-objective optimization problems, PSO algorithms have been developed to solve constrained problems, multi-objective optimization problems, problems with dynamically changing landscapes, and to find multiple solutions. For a review, see (Engelbrecht 2005).

A number of research directions are currently pursued, including:

- Theoretical aspects
- Matching algorithms (or algorithmic components) to problems
- Application to more and/or different class of problems (e.g., multiobjective)
- Parameter selection
- Comparisons between PSO variants and other algorithms
- New variants

Notes

[1]Without loss of generality, the presentation considers only minimization problems.

References

- M. Clerc. *Particle Swarm Optimization*. ISTE, London, UK, 2006.
- M. Clerc and J. Kennedy. The particle swarm-explosion, stability and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58-73, 2002.
- A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, Chichester, UK, 2005.
- F. Heppner and U. Grenander. A stochastic nonlinear model for coordinated bird flocks. *The Ubiquity of Chaos*. AAAS Publications, Washington, DC, 1990.
- J. Kennedy. Bare bones particle swarms. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 80-87, IEEE Press, Piscataway, NJ, 2003.
- J. Kennedy. Swarm Intelligence. In *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*. A. Y. Zomaya (Ed.) , pages 187-219, Springer US, Secaucus, NJ, 2006.
- J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942-1948, IEEE Press, Piscataway, NJ, 1995.
- J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 4104-4108, IEEE Press, Piscataway, NJ, 1997.
- J. Kennedy, and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA, 2001.
- R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8(3):204-210, 2004.
- A. Nowak, J. Szamrej, and B. Latane. From private attitude to public opinion: A dynamic theory of social impact. *Psychological Review*, 97(3):362-376, 1990.
- R. Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, Article ID 685175, 10 pages, 2008.
- R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. An overview. *Swarm Intelligence*, 1(1):33-57, 2007.
- W. T. Reeves. Particle systems--A technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91-108, 1983.
- C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *ACM Computer Graphics*, 21(4):25-34, 1987.

Internal references

- Marco Dorigo (2007) Ant colony optimization. Scholarpedia, 2(3):1461.
- Howard Eichenbaum (2008) Memory. Scholarpedia, 3(3):1747.
- Marco Dorigo and Mauro Birattari (2007) Swarm intelligence. Scholarpedia, 2(9):1462.
- Arkady Pikovsky and Michael Rosenblum (2007) Synchronization. Scholarpedia, 2(12):1459.

External Links

- Papers on PSO are published regularly in many journals and conferences:
  - Swarm Intelligence (http://www.springer.com/11721) (whose main objective is to archive the most important advancements in the swarm intelligence field) regularly publishes articles on PSO. Other journals also publish articles about PSO. These include the IEEE Transactions series, Applied Soft Computing (http://www.elsevier.com/locate/asoc/) , Natural Computing (http://www.springer.com/computer/foundations/journal/11047) , Structural and Multidisciplinary Optimization (http://www.springer.com/engineering/journal/158) , and others.
  - *ANTS - International Conference on Swarm Intelligence* (http://iridia.ulb.ac.be/~ants) , started in 1998.
  - *The IEEE Swarm Intelligence Symposia* (http://www.computelligence.org/sis) , started in 2003.
  - Special sessions or special tracks on PSO are organized in many conferences. Examples are the IEEE Congress on Evolutionary Computation (CEC) and the Genetic and Evolutionary Computation (GECCO) series of conferences.
  - Papers on PSO are also published in the proceedings of many other conferences such as Parallel Problem Solving from Nature conferences, the European Workshops on the Applications of Evolutionary Computation and many others.

See also

Swarm Intelligence, Ant Colony Optimization, Optimization, Stochastic Optimization

Marco Dorigo, Marco A. Montes de Oca, Andries Engelbrecht (2008) Particle swarm optimization. Scholarpedia, 3(11):1486, (go to the first approved version)
Created: 9 May 2006, reviewed: 21 November 2008, accepted: 21 November 2008

Invited by: Dr. Eugene M. Izhikevich, Editor-in-Chief of Scholarpedia, the peer-reviewed open-access encyclopedia
Action editor: Dr. Eugene M. Izhikevich, Editor-in-Chief of Scholarpedia, the peer-reviewed open-access encyclopedia
Reviewer A: Dr. Martin Middendorf, Universität Leipzig, Leipzig, Germany
Retrieved from "http://www.scholarpedia.org/article/Particle_swarm_optimization"

Categories: Computational Intelligence | Artificial Intelligence | Artificial Life